НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 004.912

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Інформаційна підтримка розпізнавання документів для мобільної платформи*»

Виконав: студент *4* курсу, групи *ІС-52*
Камінський Андрій Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Олійник Ю.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *старший викладач Халус О.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Кисленко Ю.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Камінський А.В. _____
(підпис)

Київ – 2019 р.

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)

“___”_____2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Камінському Андрію Володимировичу
(прізвище, ім`я, по батькові)

1. Тема проекту «*Інформаційна підтримка розпізнавання документів для мобільної платформи»*

керівник проекту *Олійник Юрій Олександрович, ст. викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна діяльності

2. Схема структурна варіантів використання

3. Рішення з математичного забезпечення

4. Схема структурна класів програмного забезпечення

5. Схема структурна послідовності

6. Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«23» квітня 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*16.04.2018*	
2.	*Аналіз існуючих методів розв'язання задачі*	*19.04.2018*	
3.	*Постановка та формалізація задачі*	*20.04.2018*	
4.	*Розробка інформаційного забезпечення*	*23.04.2018*	
5.	*Алгоритмізація задачі*	*25.04.2018*	
6.	*Обґрунтування використовуваних технічних засобів*	*28.04.2018*	
7.	*Розробка програмного забезпечення*	*01.05.2018*	
8.	*Налагодження програми*	*06.05.2018*	
9.	*Виконання графічних документів*	*10.05.2018*	
10.	*Оформлення пояснювальної записки*	*15.05.2018*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ А.В. Камінський
(підпис)

Керівник проекту _____ Ю.О. Олійник
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5212.1181-с.ПЗ	Пояснювальна записка	74	
2	А4	ДП ІС-5212.1181-с.ТЗ	Технічне завдання	10	
3	А4	ДП ІС-5212.1181-с.ПМВ	Програма та методика випробувань	12	
			Матеріали графічні		
4	А4	ДП ІС-5212.1181-с.ССД	Схема структурна діяльності	1	
5	А4	ДП ІС-5212.1181-с.ССВ	Схема структурна варіантів		
			використання	1	
6	А3		Рішення з математичного забезпечення	1	
7	А3	ДП ІС-5212.1181-с.ССК	Схема структурна класів програмного		
			забезпечення	1	
9	А3	ДП ІС-5212.1181-с.ССП	Схема структурна послідовності	1	
10	А4	ДП ІС-5212.1181-с.ССК	Схема структурна компонентів		
			програмного забезпечення	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5212.1181-с.ВДП		
Розробн.		Камінський А.В.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Олійник Ю.О.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: «Інформаційна підтримка розпізнавання документів для мобільної платформи»

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 10 рисунків, 11 таблиць, 1 додаток, 16 джерел.

Дипломний проект присвячений інформаційній підтримці розпізнавання документів для мобільної платформи з метою категоризації документів і спрощення ведення документообігу.

В дипломному проекті були розглянуті методи визначення категорії документів, наївний баєсівський класифікатор, алгоритми та методи розпізнавання тексту з зображень.

У розділі з інформаційного забезпечення були визначені вхідні та вихідні дані до застосування, були описані приклади структур нереляційної базі даних.

У розділі математичного забезпечення обґрунтований та описаний наївний баєсівський алгоритм, як обраний метод вирішення проблеми визначення категорії документів.

У розділі з програмного забезпечення описані основні засоби розробки застосування, висунуті вимоги до технічного забезпечення, обрано та обґрунтовано архітектуру програмного забезпечення.

У технологічному розділі описані інструкція користувача та методи випробування програмного продукту.

НАЇВНИЙ БАЄСОВСЬКИЙ АЛГОРИТМ, РОЗПІЗНАВАННЯ ТЕКСТУ, КЛАССИФІКАЦІЯ ДОКУМЕНТІВ, ОБРОБКА ТЕКСТУ З ЗОБРАЖЕННЯ

ABSTRACT

Structure and scope of work. Explanatory note of the diploma project consists of five sections, contains 10 drawings, 11 tables, 1 applications, 16 sources.

The diploma project is devoted to information support for document recognition for the mobile platform in order to categorize documents and simplify document management.

The diploma project examined methods for determining the category of documents, a naive Bayes classifier, algorithms and methods for recognizing text from images.

In the information support section, input and output data were identified prior to application, examples of structures of the non-relational database were described.

In the section of mathematical support, the naive Baies algorithm is justified and described, as the chosen method of solving the problem of determining the category of documents.

The software section describes the main tools for developing the application, the requirements for the technical support, the software architecture is selected and substantiated.

The technology section describes the user manual and test methods for the software product.

NAÏVE BAIES ALGORUTHM, TEXT RECOGNITION, DOCUMENT CLASSIFICATION, TEXT PROCESSING FROM IMAGE

ЗМІСТ

ВСТУП

На сьогоднішній день майже кожна сфера життя все більше прямує до автоматизації та комп’ютеризації усіх монотонних процесів. Спрощення роботи сучасної людини є головною метою технічного прогресу. Але нажаль далеко не всі змогли пристосуватися до нововведень, особливо якщо це стосується роботи з документами, анкетами, виписками, тощо. Попри наявність безлічі електронних ресурсів, спеціалісти все ще працюють із паперовими документами, нехтуючи всіма перевагами інформаційних технологій.

Не варто забувати і про те, що гостро стоїть проблема так би мовити «синхронізації» старих паперових документів із сучасними електронними базами даних. Важко перейти на нову систему документообігу, коли для цього треба вручну перенести величезну кількість інформації. А робота одночасно з паперовими та електронними документами робить весь процес не тільки неефективним, а ще й дуже часозатратним [1].

Розробка інформаційної підтримки розпізнавання документів для мобільної платформи допоможе значно поліпшити процес ведення документообігу, мінімізує можливі помилки та неточності при внесені даних, прискорить та полегшить процеси пошуку, редагування та видачі документів.

Основною функцією застосування є класифікація типу документу для його розпізнавання та подальшої обробки, враховуючи його тип та шаблон. Алгоритм застосування буде використовувати наївний баєсівський класифікатор, який буде категоризувати документ за розпізнаними та виділеними атрибутами. Також застосування надасть користувачу змогу зберегти опрацьований текст у вигляді структури даних, яку можна буде додати у базу даних для подальшого зберігання та обробки.

Таким чином застосування надає можливість створити єдиний реєстр необхідної інформації для автоматизованого документообігу. Розроблений

програмний продукт слугуватиме помічником для менеджерів по роботі з інформацією, спеціалістам з документообігу надаючи їм можливість централізування та об’єднання документів, шляхом переведення інформації на папері у електронний вигляд.

Дипломний проект присвячений інформаційній підтримці розпізнавання документів для мобільної платформи.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

У сучасному цифровому столітті все більше і більше компаній переходить на електронний документообіг, що дозволяє позбутися від паперу і прискорити процеси прийняття рішень. Але на даний момент повністю відмовитися від нього неможливо, оскільки організації зобов'язані зберігати архіви документів в паперовому вигляді, ряд бізнес-процесів в обов'язковому порядку вимагає наявність паперових документів. Близько 80% всіх ділових документів (наприклад, рахунки, сертифікати, договори і т.д.) складають слабоструктуровані документи – документи з нечіткою структурою, яка може змінюватися від організації до організації. Заповнення даних про таких документах в системах електронного документообігу часто робиться вручну і займає чимало часу. Для зменшення часу внесення даних про паперові документи необхідно розробити метод обробки слабоструктурованих паперових документів.

Розроблені методи обробки не враховують проблеми різної структури документів одного типу, помилок в даних, помилок розпізнавання тексту. Тому необхідно розробити метод виділення атрибутів з урахуванням даних особливостей.

Можна виділити два типи атрибутів, важливих для подальшого використання: поодинокі атрибути (наприклад, номер, дата документа), множинні атрибути (таблиці документів). Стандартні етапи перенесення даних з паперових документів в системи електронного документообігу: сканування, розпізнавання, класифікація, витяг атрибутів, завантаження в базу даних. Завдання сканування і розпізнавання виносяться за рамки даного дослідження [2].

Коли задача класифікації типу документа вирішена, необхідно витягти атрибути. Для вирішення цього завдання скористаємося результатом того,

що було досягнуто на попередньому етапі. Якщо вирішена задача класифікації, то це дає можливість визначити перелік атрибутів, які повинні бути вилучені.

1.1.1 Опис процесу діяльності

Об’єктом автоматизації є інформаційна підтримка розпізнавання документів для мобільної платформи.

Класифікація документів – це поділ документів на класи за найбільш загальними ознаками схожості та відмінності.

Метою класифікації документів є підвищення оперативності роботи апарату управління та відповідальності виконавців. Первинна класифікація документів забезпечує їх швидкий пошук, підвищує оперативність роботи з ними, прискорює виконання і контроль.

Будь-яка компанія щодня стикається з необхідністю обробки десятків, а то й сотень різних документів. Їх класифікація стає одним з найважливіших процесів у вхідному, вихідному і внутрішньому документообігу. І чим більше потік документів, тим більше часу і трудовитрат потрібно на те, щоб його обробити. А якщо додати до «класичним» документам ще електронні листи і запити, які також вимагають класифікації і аналізу, то завдання ускладнюється в рази.

Класифікація документів за змістом, або семантична класифікація, – це той випадок, коли для розуміння того, до якої групи належить документ, необхідно спочатку його прочитати. Наприклад, щоб зрозуміти, що у вас в руках саме договір, а не додаток до нього, необхідно вивчити хоча б перший абзац. При цьому кожен документ може бути одночасно класифікований і по типу, і за змістом. Співробітникам, відповідальним за обробку, необхідно визначити тип документа, перевести паперовий документ в електронний, ввести його в систему, а при необхідності направити до відповідного підрозділу. Ручна обробка документів нерідко призводить до появи помилок

введення, спотворення даних, що абсолютно неприпустимо, якщо компанія хоче займати гідне місце на ринку [3].

При відкритті додатку користувач зможе одразу перейти до процесу розпізнавання документів. Для цього йому потрібно буде завантажити фотографію документу з галереї, або зробити її з камери. При цьому треба враховувати ряд вимог до фотографії, які будуть наведені нижче, для підвищення вірогідності якісного розпізнавання.

Коли користувач визначиться з зображенням, йому потрібно буде натиснути всього одну кнопку, і застосування почне процес класифікації документу. Тоді, коли документ буде віднесено до певної категорії, застосування виділить атрибути, змість яких буде внесено у відповідну таблицю бази даних.

Після завершення роботи основного алгоритму застосування, користувач зможе подивитися звіт про розпізнаний документ, вибравши його зі списку в головному меню.

Розглянемо детальніше процес виконання операцій застосування за допомогою схеми структурної діяльності, що наведена у частині графічного матеріалу.

1.1.2 Опис функціональної моделі

Перед проектуванням діаграми необхідно виділити дійових осіб застосування (акторів), а потім визначити дії, які може виконувати кожен з акторів.

Актором системи є лише користувач, оскільки система не потребує адміністрування.

Користувач. Незареєстрована особа, яка може працювати з функціями застосування, а саме фотографувати документи для розпізнавання, завантажувати зображення, переглядати розпізнаний текст, визначати категорію документу на основі розпізнаного тексту зображення,

зберігат результат обробки у локальну базу даних та переглядати історію помилок. Також користувач може переглянути інструкцію з користування додатком та обмеження на завантажене зображення.

Визначимо, які функції актор виконує в системі, для цього наведемо таблицю 1.1, в якій описані актори, функції та їх описи.

Таблиця 1.1 – Опис функцій застосування

Актор	Функція	Опис функції
Користувач	Фотофіксація документу	Користувач надає системі інформацію (фотографує документ), саме цю інформацію (зображення). Буде аналізувати система
Користувач	Завантаження зображення	Після фотографування документу, система надає користувачу можливість зберегти зображення або сфотографувати ще раз, якщо зображення вийшло нечітким.
Користувач	Перегляд обробленого тексту	Система надає користувачу можливість переглянути результат обробки, щоб користувач перевірив якість обробленого тексту
Користувач	Збереження результату обробки	Система надає можливість користувачу зберегти розпізнану інформацію у локальну базу даних.
Користувач	Перегляд історії помилок	Система надає можливість користувачу переглядати список документів, які не були розпізнані.

Для специфікації функціональної поведінки застосування побудована структурна схема варіантів використання, що наведена в частині графічного матеріалу.

1.2 Огляд наявних аналогів

Основною функцією застосування є розпізнавання тексту з зображення. На даний момент часу існує досить велика кількість програм, які працюють з розпізнаванням тексту, що забезпечує вирішення ряду наукових та прикладних задач при ідентифікації об'єктів різної природи.

Існує декілька основних методів розпізнавання тексту.

Шаблонний метод

Цей метод базується на перетворенні частини зображення із символом у растрове зображення. Потім потрібно порівняти його із наявними у базі шаблонами і визначити зображення, яке менше за всіх відрізняється від нашого. Безсумнівним плюсом такого методу є стійкість до нечітких зображень. Тобто з великою вірогідністю саме цей метод найбільш успішно розпізнає нечітке зображення. Також такий метод має велику швидкість обробки даних. Проте таким методом можна розпізнати тільки ті документи, шрифти яких відомі методу. В противному випадку шаблонний метод буде неефективним навіть при роботи з дуже чіткою картинкою [4].

Структурний метод

Структурні методи розпізнавання зберігають інформацію не про поточкове написання символу, а про його топологію. Еталон містить інформацію про взаємне розташування окремих складових частин символу. Перевага методу – стійкість до зсуву і повороту символу на невеликий кут, до різних стильових варіацій шрифтів. Однак, при повороті на кут, більший десяти градусів, даний метод не може бути використаний для розпізнавання символів. При застосування цього методу неважливими стають такі ознаки як розмір букви, що розпізнається і навіть шрифт, яким вона надрукована.

Проте, основною проблемою цього методу є ідентифікація знаків, які містять певні дефекти (наприклад, розрив ліній або з'єднання сусідніх ліній).

Ознаковий метод

Ознакові методи базуються на тому, що зображенню ставиться у відповідність N-мірний вектор ознак. Розпізнавання полягає в порівнянні вектора ознак з набором еталонних векторів тієї ж розмірності. Переваги методу – простота реалізації, хороша узагальнююча здатність, висока швидкість розпізнавання. Недолік методу – висока чутливість до дефектів зображення. Крім того, ознакові методи мають інший недолік — на етапі виділяння ознак відбувається незворотня втрата частини інформації про символ. Виділяння ознак проходить незалежно, тому інформація про взаємне розташування елементів символів втрачається.

Оптичне розпізнавання тексту (англ. optical character recognition, OCR) – це механічне або електронне переведення збереженого рукописного, машинописного або друкованого тексту в послідовність кодів, що використовують для представлення в текстовому редакторі. Системи оптичного розпізнавання тексту вимагають калібрування для роботи з конкретним шрифтом, тобто вони відносяться саме до шаблонних методів розпізнавання [5].

Розпізнавання символів он-лайн іноді плутають з оптичним розпізнаванням символів. Метод оптичного розпізнавання символів – офф-лайн-метод, що працює зі статичною формою подання тексту, у той час як он-лайн-розпізнавання символів ураховує рухи під час писання.

На сьогоднішній день існує дуже багато аналогічних продуктів, таких як Abby FlexiCapture, SimpleOCR, OmniPage, Readiris, CuneiForm, Tesseract, тощо. Зупинимось детальніше на двох із них.

Основне призначення ABBYY FlexiCapture – витягати з документів певні дані і заносити їх в інформаційну систему. Це рішення для потокового

введення даних і документів. Продукт автоматизує вилучення інформації з паперових документів і зберігає дані в інформаційній системі підприємства. ABBYY FlexiCapture дозволяє різним організаціям, в тому числі великим корпораціям, урядовим структурам і освітнім установам, автоматизувати процес введення даних в інформаційні системи, знизити витрати і підвищити якість обслуговування клієнтів [6].

Tesseract – вільна програма для розпізнавання текстів, що розроблялася Hewlett-Packard з середини 1980-х по середину 1990-х, а потім 10 років «пролежала на полиці». У серпні 2006 року Google купив її і відкрив вихідні тексти під ліцензією Apache 2.0 для продовження розробки. На даний момент програма вже працює з UTF-8, підтримка мов здійснюється за допомогою додаткових модулів [7].

Хоча вже існують тисячі застосувань і способів використання розпізнавання тексту, треба зазначити, що даний програмний продукт все ж унікальний, і ось чому. По-перше він планується як безкоштовний, що дуже важливо для потреби звичайного користувача, або коли проект потрібно просто спочатку протестувати, без попереднього вкладання великих грошей. По-друге, основною ідеєю є не тільки розпізнавання, але й класифікація самого документу за основними його атрибутами. І вже потім застосування зможе витягнути потрібну інформацію для подальшого збереження у базі даних. Класифікація документу за допомогою наївного баєсовського алгоритму зможе зробити процес розпізнавання більш швидким та перевести в електронний формат велику кількість інформації без зусиль з боку користувача.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням розробки є інформаційна підтримка розпізнавання документів для мобільної платформи, для реалізації категоризації документу що розпізнається.

1.3.2 Цілі та задачі розробки

Головною метою розробки є спрощення процесу розпізнавання документів для переведення інформації у електронний формат.

Цілями розробки є:

- полегшення процесу ведення документообігу;
- спрощення розпізнавання однотипних документів.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- розробити функцію для розпізнавання тексту;
- розробити функцію класифікації документів;
- розробити функцію переведення тексту у структуру;
- розробити базу даних для збереження розпізнаної інформації;
- розробити функцію трекінгу помилок при обробці.

Висновок до розділу

У даному розділі описане предметне середовище, був наведений опис процесу діяльності, описані ролі актору застосування та дії, які може виконувати користувач, сформульовані призначення. Наведені цілі та задачі розробки та варіанти використання.

Описані наявні системи зі схожими функціями. Аналоги проаналізовано та порівняно виходячи з функціональних можливостей програми. Був зроблений висновок, що проаналізовані аналоги не відповідають меті розробки, адже всі вони не розраховані для в першу чергу категоризації документів для подальшого переведення інформації у електронний формат.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані в застосування надходять від користувача. Дані поступають у вигляді фотографії документу, що буде розпізнаватися. Зображення робиться за допомогою камери мобільного телефону або завантажується з галереї.

Далі наведено опис вхідних даних, які вносить користувач. Оскільки обробка фотографії є досить специфічною задачею, на зображення накладається ряд обмежень з метою покращення коефіцієнту розпізнавання. Тобто мінімізації кількості помилок при обробці зображення.

Вхідними даними для системи є:

- Зображення документу.

До вхідних даних є певні вимоги:

- Зображення повинно бути чітким;
- Зображення повинно бути контрастним;
- Зображення має бути цілісним (не допускаються обрізані документи);
- Текст на зображенні має іти в звичайному порядку (по горизонталі). Перевернуті зображення, зображення з дзеркальним порядком тексту не допускаються.

Таким чином застосуванню на вхід подається зображення документу, з якого потім буде виділена інформація.

2.2 Вихідні дані

Вихідними даними застосування є оброблений текст документу, який користувач може попередньо переглянути на наявність помилок. Також на вихід подається запис з обробленою інформацією у локальну базу даних,

якщо користувач підтвердить розпізнаний текст. І нарешті, користувач зможе переглянути історію помилок та нерозпізнаних документів.

Вихідними даними системи є:

- розпізнаний текст для попереднього перегляду;
- новий запис з текстом у базі даних;
- перелік помилок та нерозпізнаних документів.

Таким чином, на основі даних, які надходять від користувача, формуються розпізнані документи, їх категорії, заповнюються таблиці в базі даних та звіти про помилки.

2.3 Опис структури бази даних

При проектуванні бази даних застосування було вирішено, що для кожного типу документу потрібно створити окрему сутність добавляти сутності при збільшенні кількості типів документів, які може розпізнати додаток. Так як зв’язок між сутностями відсутній, було вирішено обрати нереляційну базу даних. На даний момент додаток вміє розпізнавати 2 типи документів «Заява про перехід на спрощену систему оподаткування» та «Реєстраційна картка АЦСК», відповідно БД містить однойменні сутності.

Для створення системи використовується нереляційна база даних Realm.

Сутність «Заява про перехід на спрощену систему оподаткування» – описує наявні атрибути заяви, які заповнюються вручну, та будуть розпізнані застосуванням для подальшого перенесення у базу (таблиця 2.1). Перерахуємо основний набір властивостей сутності «Заява про перехід на спрощену систему оподаткування»:

- Найменування органу податкової служби;
- Тип особи;
- Код ЄДПОУ;
- Найменування суб’єкта господарювання;

- Група;
- Податкова ставка;
- Адреса ведення діяльності;
- Коди ведення діяльності;
- Дата переходу;
- Контактний телефон;

Сутність «Реєстраційна картка АЦСК» – описує наявні реєстраційної картки, які заповнюються вручну, та будуть розпізнані застосуванням для подальшого перенесення у базу (таблиця 2.2). Перерахуємо основний набір властивостей сутності «Реєстраційна картка АЦСК»:

- Прізвище;
- Ім’я;
- По батькові;
- Номер картки платника податку;
- Місце проживання;
- Телефон;
- Електрона пошта;
- Ключова фраза;
- Дата заповнення;

Висновок до розділу

У даному розділі описані вхідні дані, які застосування отримує від користувача, та вихідні дані, які подаються після обробки вхідних даних, тобто розпізнавання тексту. Процес розпізнавання та категоризації документів використовує дані від користувача, тобто фотографію документу для подальшої обробки та забезпечує коректні результати для актуального типу документу індивідуально для кожного з користувачів.

Оскільки у застосуванні використовується нереляційна база даних, таблиці якої відповідають структурі документу, що розпізнається, у розділі були наведені приклади таблиць для двох типів документів. Описано структури таблиць бази даних з детальною інформацією по кожній таблиці, з детальним змістовим описом полів. Для роботи застосуванню інформаційної підтримки розпізнавання документів для мобільної платформи потрібна лише по одній таблиці для кожного типу документу.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Застосування повинно за потребою користувача розпізнавати сфотографований документ. Для збільшення точності розпізнавання та виділення саме тих даних, які потрібно буде перенести у локальну базу даних, спочатку потрібно зрозуміти, який саме документ перед нами та які його атрибути. Під атрибутами документу мається на увазі ідентифікована (іменована) характеристика частини реквізиту. Тобто це фіксовані слова або найменування, після яких клієнт має вписати свою унікальну інформацію. Наприклад ПІБ, адреса, номер ідентифікаційного коду – все це є атрибутами документу.

У даному випадку у ролі користувача виступають бухгалтери, менеджери, аналітики, співробітники державних установ. Всі ті, хто повинен працювати з великими обсягами паперових документів, заяв, анкет, тощо. Їх мета – якомога швидше та ефективніше працювати із документами, виконувати пошук та редагування, видачу самих документів. Все це потребує спрощення системи документообігу, а найкращий спосіб це зробити – цілком перейти на електронну систему ведення документів [8].

Для реалізації роботи системи електронного документообігу, звичайно ж, необхідно у тому ж числі перенести уже наявні паперові документи у електронну базу даних. Таким чином перед нами стає задача розпізнавання тексту. Але, оскільки система буде працювати не просто з текстом, а документами, які бувають багатьох типів, логічно зазначити, що головною задачею, що буде вирішувати застосування, є задача класифікації.

Класифікація документів буде здійснюватися за допомогою наївного байєсівського алгоритму. Застосування буде використовувати «еталонні» документи, тобто приклади того, як має виглядати документ того чи іншого типу. На основі таких шаблонів, алгоритм класифікації буде визначати тип

документу, витягувати пари ключ-значення згідно його атрибутів, та записувати ці значення у потрібну таблицю бази даних.

3.2 Математична постановка задачі

Призначенням цієї задачі є визначити, до якого типу належить документ.

Поставлену задачу формально можна відобразити наступним чином.

Дано:

- *d* – документ, що розпізнається;
- *c* – клас, до якого може належати документ;
- *P(c|d)* – ймовірність що документ *d* належить класу *c*;
- *P(d|c)* – ймовірність зустріти документ *d* серед всіх документів класу *c*;
- *P(c)* – безумовна ймовірність зустріти документ класу *c* в корпусі документів;
- *P(d)* – безумовна ймовірність документа *d* в корпусі документів.

Необхідно:

- розрахувати ймовірність *P(c|d)* того, що документ належить до класу для кожного з класів;
- за формулою визначити клас з максимальною ймовірністю належності;
- визначити, до якого класу належить документ, що розпізнається.

3.3 Обґрунтування методу розв’язання

Задача, описана у пункті 3.2, може бути віднесена до такого методу класифікації документів, як наївний байєсів класифікатор.

В основі наївного баєсівського клафікатора NBC (Naïve Bayes Classifier) лежить теорема Байєса. Теорема Байєса одна з основних теорем елементарної теорії ймовірностей, яка дозволяє визначити ймовірність

певної події за умови, що сталася інша статистично взаємозалежна з нею подія. Іншими словами, за формулою Байеса можна більш точно перелічити ймовірність, взявши до уваги як раніше відому інформацію, так і дані нових спостережень.

Наївний байєсівский алгоритм (НБА) – це алгоритм класифікації, заснований на теоремі Байеса з припущенням про незалежність ознак. Іншими словами, НБА передбачає, що наявність якої-небудь ознаки в класі не пов’язана з наявністю будь-якої іншої ознаки. Наприклад, фрукт може вважатися апельсином, якщо він помаранчевий, круглий і його діаметр становить близько 10 сантиметрів. Навіть якщо ці ознаки залежать один від одної або від інших ознак, в будь-якому випадку вони вносять незалежний внесок у ймовірність того, що цей фрукт є апельсином. У зв’язку з таким припущенням алгоритм називається «наївним» [9].

Моделі на основі НБА досить прості і вкрай корисні при роботі з дуже великими наборами даних. При своїй простоті НБА здатний перевершити навіть деякі складні алгоритми класифікації.

Основні **переваги** наївного байєсівського класифікатора - простота реалізації і низькі обчислювальні витрати при навчанні та класифікації. У тих рідкісних випадках, коли ознаки дійсно незалежні (або майже незалежні), наївний байєсівский класифікатор (майже) оптимальний.

Основний його **недолік** – відносно низька якість класифікації в більшості реальних задач.

Залежно від точної природи ймовірнісної моделі, наївні байєсівські класифікатори можуть навчатися дуже ефективно. У багатьох практичних додатках для оцінки параметрів для наївних Байесови моделей використовують метод максимальної правдоподібності; іншими словами, можна працювати з наївною байєсівської моделлю, не вірячи в Байєсову ймовірність і не використовуючи байєсовські методи.

Незважаючи на наївний вигляд і, безсумнівно, дуже спрощені умови, наївні байєсовські класифікатори часто працюють набагато краще в багатьох складних життєвих ситуаціях.

Перевагою наївного байєсівського класифікатора є мала кількість даних необхідних для навчання, оцінки параметрів і класифікації.

Позитивні і негативні сторони наївного байєсівського алгоритму

Позитивні сторони.

Класифікація, в тому числі багатокласова, виконується легко і швидко.

Коли допущення про незалежність виконується, НБА перевершує інші алгоритми, такі як логістична регресія (logistic regression), і при цьому вимагає менший обсяг навчальних даних.

НБА краще працює з категорійними ознаками, ніж з неперервними. Для неперервних ознак передбачається нормальний розподіл, що є досить сильним допущенням.

Негативні сторони.

Якщо в тестовому наборі даних є певне значення категорійного ознаки, яке не зустрічалося в навчальному наборі даних, тоді модель присвоїть нульову ймовірність цього значення і не зможе зробити прогноз. Це явище відоме під назвою «нульова частота» (zero frequency). Дану проблему можна вирішити за допомогою згладжування. Одним з найпростіших методів є згладжування по Лапласу (Laplace smoothing).

Ще одним обмеженням НБА є припущення про незалежність ознак. В реальності набори повністю незалежних ознак зустрічаються вкрай рідко.

Практичне застосування байєсівського алгоритму.

Класифікація в режимі реального часу. НБА дуже швидко навчається, тому його можна використовувати для обробки даних в режимі реального часу.

Багатокласова класифікація. НБА забезпечує можливість багатокласової класифікації. Це дозволяє прогнозувати ймовірність для безлічі значень цільової змінної.

Класифікація текстів, фільтрація спаму, аналіз тональності тексту. При вирішенні завдань, пов'язаних з класифікацією текстів, НБА перевершує багато інших алгоритми. Завдяки цьому, даний алгоритм знаходить широке застосування в області фільтрації спаму (ідентифікація спаму в електронних листах) і аналізу тональності тексту (аналіз соціальних медіа, ідентифікація позитивних та негативних думок клієнтів).

Рекомендаційні системи. Наївний байєсівский класифікатор в поєднанні з коллаборативной фільтрацією (collaborative filtering) дозволяє реалізувати рекомендаційну систему. В рамках такої системи за допомогою методів машинного навчання та інтелектуального аналізу даних нова для користувача інформація фільтрується на підставі прогнозованої думки цього користувача про неї [10].

3.4 Опис методів розв’язання

Наївний байєсівський класифікатор об'єднує модель з правилом рішення. Одне загальне правило має вибрати найбільш ймовірну гіпотезу; воно відоме як апостеріорне правило прийняття рішення. Оцінка максимуму апостеріорної імовірності (Maximum a posteriori probability estimate, MAP) у баєсівій статистиці – це мода апостеріорного розподілу. Вона може застосовуватися для отримання точкової оцінки неспостережуваної величини на базі емпіричних даних.

Теорема Байєса, що лежить в основі наївного байєсівського алгоритму має наступний вигляд:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{3.1}$$

Тут *P(c|d)* – ймовірність що документ *d* належить класу *c*, саме її нам треба розрахувати;

P(d|c) – ймовірність зустріти документ *d* серед всіх документів класу *c*;

P(c) – безумовна ймовірність зустріти документ класу *c* в корпусі документів;

P(d) – безумовна ймовірність документа *d* в корпусі документів.

Її сенс на доступному рівні можна виразити таким чином. Теорема Байєса дозволяє переставити місцями причину і наслідок. Знаючи з якою ймовірністю причина призводить до якоїсь події, ця теорема дозволяє розрахувати ймовірність того що саме ця причина призвела до нинішньої події.

Мета класифікації полягає в тому, щоб зрозуміти до якого класу належить документ, тому нам потрібна не сама ймовірність, а найбільш ймовірний клас. Байєсівський класифікатор використовує оцінку апостеріорного максимуму для визначення найбільш вірогідного класу. Грубо кажучи, це клас з максимальною ймовірністю.

$$c_{map} = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} \tag{3.2}$$

Тобто нам треба розрахувати ймовірність для всіх класів і вибрати той клас, який має максимальну вірогідність. Зверніть увагу, знаменник (ймовірність документа) є константою і ніяк не може вплинути на ранжування класів, тому в нашому завданні ми можемо його ігнорувати.

$$c_{map} = \arg\max_{c \in C} [P(d|c)P(c)] \tag{3.3}$$

Далі робиться припущення яке і пояснює чому цей алгоритм називають наївним. Припущення умовної незалежності. У живій мові ймовірність появи слова сильно залежить від контексту. Байєсівський ж класифікатор документ як набір слів ймовірності яких умовно не залежать

одне від одного. Виходячи з цього припущення умовна ймовірність документа апроксимується твором умовних ймовірностей всіх слів входять до документ.

$$P(d|c) \approx P(w_1|c)P(w_2|c)\dots P(w_n|c) = \prod_{i=1}^{n} P(w_i|c) \tag{3.4}$$

Цей підхід також називається Unigram Language Model. Мовні моделі грають дуже важливу роль в задачах обробки натуральних мов, але виходять за межі цієї замітки.

Підставивши отриманий вираз в формулу 3.3 ми отримаємо:

$$c_{map} = \arg\max_{c \in C} \left[P(c) \prod_{i=1}^{n} P(w_i|c) \right] \tag{3.5}$$

Але можлива проблема арифметичного переповнення. При досить великій довжині документа доведеться множити велику кількість дуже маленьких чисел. Для того щоб при цьому уникнути арифметичного переповнення знизу часто користуються властивістю логарифма добутку *log(a*b) = log(a) + log(b)*. Так як логарифмічна функція монотонна, її застосування до обох частин виразу змінить тільки його чисельне значення, але не параметри при яких досягається максимум. При цьому, логарифм від числа близького до нуля буде числом негативним, але в абсолютному значенні істотно більшим ніж вихідне число, що робить логарифмічні значення ймовірностей більш зручними для аналізу. Тому, ми переписуємо нашу формулу 3.3 з використанням логарифма.

$$c_{map} = \arg\max_{c \in C} \left[\log P(c) + \sum_{i=1}^{n} \log P(w_i|c) \right] \tag{3.6}$$

Основа логарифму в даному випадку не має значення. Ви можете використовувати як натуральний, так і будь-який інший логарифм.

Оцінка параметрів Байєсівської моделі.

Оцінка ймовірностей $P(c)$ і $P(w_i|c)$ здійснюється на навчальній вибірці.

Імовірність класу ми можемо оцінити як: $P(c) = \frac{D_c}{D}$ де D_c – кількість документів, які належать класу c, а D – загальна кількість документів в навчальній вибірці.

Оцінка ймовірності слова в класі може робитися декількома шляхами. Тут наведений multinomial bayes model.

$$P(w_i|c) = \frac{W_{ic}}{\sum_{i' \in V} W_{i'c}} \tag{3.7}$$

W_{ic} – кількість разів скільки i-те слово зустрічається в документах класу c;
V – словник корпусу документів (список всіх унікальних слів).
Іншими словами, чисельник описує скільки разів слово зустрічається в документах класу (включаючи повтори), а знаменник – це сумарна кількість слів у всіх документах цього класу.

Проблема невідомих слів.

З формулою 3.7 є одна невелика проблема. Якщо на етапі класифікації вам зустрінеться слово якого ви не бачили на етапі навчання, то значення W_{ic}, а слідом і $P(w_i|c)$ дорівнюватимуть нулю. Це призведе до того, що документ з цим словом не можна буде класифікувати, так як він буде мати нульову ймовірність по всіх класах. Позбутися від цієї проблеми шляхом аналізу більшої кількості документів не вийде. Ви ніколи не зможете скласти навчальну вибірку, яка містить всі можливі слова, включаючи неологізми, друкарські помилки, синоніми і т.д. Типовим рішенням проблеми невідомих слів є аддитивное згладжування (згладжування Лапласа). Ідея полягає в тому що ми вдаємо начебто бачили кожне слово на один раз більше, тобто додаємо одиницю до частоти кожного слова [11].

$$P(w_i|c) = \frac{W_{ic}+1}{\sum_{i' \in V}(W_{i'c}+1)} = \frac{W_{ic}+1}{|V| + \sum_{i' \in V} W_{i'c}} \tag{3.8}$$

Логічно даний підхід зміщує оцінку ймовірностей в сторону менш вірогідних результатів. Таким чином, слова які ми не бачили на етапі навчання моделі отримують нехай маленьку, але все-таки не нульову ймовірність.

Підставивши обрані нами оцінки в формулу 3.6 ми отримуємо остаточну формулу за якою відбувається байєсівську класифікація.

$$c_{map} = \arg\max_{c \in C} \left[\log \frac{D_c}{D} + \sum_{i=1}^{n} \log \frac{W_{ic}+1}{|V| + \sum_{i' \in V} W_{i'c}} \right] \tag{3.9}$$

Реалізація класифікатора.

Для реалізації Байєсівського класифікатора нам необхідна навчальна вибірка в якій проставлені відповідності між текстовими документами і їх класами. Потім нам необхідно зібрати наступну статистику з вибірки, яка буде використовуватися на етапі класифікації:

- відносні частоти класів в корпусі документів. Тобто, як часто зустрічаються документи того чи іншого класу;
- сумарна кількість слів у документах кожного класу;
- відносні частоти слів у межах кожного класу;
- розмір словника вибірки. Кількість унікальних слів у вибірці.

Сукупність цієї інформації ми будемо називати моделлю класифікатора. Потім на етапі класифікації необхідно для кожного класу розрахувати значення наступного виразу і вибрати клас з максимальним значенням.

$$\log \frac{D_c}{D} + \sum_{i \in Q} \log \frac{W_{ic}+1}{|V| + L_c} \tag{3.10}$$

D_c – кількість документів в навчальній вибірці належать класу c;

D – загальна кількість документів в навчальній вибірці;

$|V|$ – кількість унікальних слів у всіх документах навчальної вибірки;

L_c – сумарна кількість слів у документах класу c в навчальній вибірці;

W_{ic} – скільки разів i-те слово зустрічалося в документах класу c в навчальній вибірці;

Q – безліч слів документу, що класифікується (включаючи повтори).

Узагальнимо описаний вище метод у вигляді покрокового алгоритму:

КРОК 1. Визначити передумови для навчання класифікатора

КРОК 2. Обчислити розріджену матрицю термінів документів для кожного класу.

КРОК 3. Обчислити частоти, з якими зустрічається кожне слово в тексті.

КРОК 4. Обчислити $P(c|d)$ згідно з формулою 3.1.

КРОК 5. Розрахувати ймовірність c_{map} для всіх класів і вибрати той клас, який має максимальну вірогідність за формулою 3.9, з урахуванням згладжування Лапласа.

Висновок до розділу

В даному розділі сформульована змістовна та математична постановки задачі класифікації документів розпізнавання для опрацювання інформації.

Визначено, що така задача може бути віднесена до такого методу класифікації документів, як наївний байєсівський класифікатор.

Для вирішення поставленої задачі обрано та описано наївний байєсівський алгоритм. Наведена оцінка параметрів з урахуванням multinomial bayes model та проблема невідомих слів, яка вирішується за допомогою згладжування Лапласа.

Реалізація класифікатора описана у вигляді формули із зміщенням оцінки імовірностей.

Описаний метод був узагальнений у вигляді покрокового алгоритму.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для створення андроїд додатку було вирішено використати мову програмування Kotlin, оскільки в травні 2017 року компанія Google сповістила про те, що з версії Android Studio 3.0 Kotlin стає офіційною інструментом розробки для ОС Android. Мова є своєрідним нащадком Java і працює на JVM (Java Virtual Machine). Також Kotlin має цілий ряд переваг, що сприяють швидшій розробці програмного продукту, з більш читабельним кодом ніж Java, та більшою гнучкістю в подальшій розробці продукту.

Переваги Kotlin над Java:

- наявність inline-функцій. Використання функцій вищого порядку тягне за собою зниження продуктивності: по-перше, функція є об'єктом, а по-друге, відбувається захоплення контексту замиканням, тобто функції стають доступні змінні, оголошені поза її тіла. А виділення пам'яті (як для об'єкта функції, так і для її класу) і віртуальні виклики займають системні ресурси. А модифікатор inline робить так що функція і лямбда, передана їй, обидві будуть вбудовані в місце виклику;
- наявність функцій-розширень. Для розширення класу новими функціями тепер не потрібно писати класи наслідники можна напряму створити функцію-розширення для даного класу;
- null-safety. Система типів в мові Kotlin націлена на те, щоб викорінити небезпеку звернення до null значень, більш відому як "Помилка на мільйон". Найпоширенішим підводним каменем багатьох мов програмування, в тому числі Java, є спроба зробити доступ до null значення. Це призводить до помилки. В Java така помилка називається NullPointerException;

- type smart cast. У багатьох випадках в Kotlin вам не потрібно використовувати явні приведення типів, тому що компілятор стежить за is-перевірками для незмінних значень і вставляє приведення автоматично, там, де вони потрібні;
- розділені інтерфейси для змінних та незмінних колекцій;
- класи даних. Іноді класи бувають необхідні тільки для зберігання деяких даних. У Kotlin такі класи називаються data-класи. Вони визначаються модифікатором data. При компіляції такого класу компілятор автоматично додає в клас функції: equals (), hashCode (), toString (), copy (). Причому при виконанні дій ці функції враховують всі властивості, які визначені в первинному конструкторі;
- ізольвані класи. Вони є, по суті, розширенням enum-класів: набір значень enum типу також обмежений, але кожна enum-константа існує тільки в єдиному екземплярі, в той час як спадкоємець ізольованого класу може мати безліч екземплярів, які можуть нести в собі якийсь стан;
- перегрузка операторів;
- співпрограми. Співпрограми спрощують асинхронне програмування, залишивши все ускладнення всередині бібліотек. Логіка програми може бути виражена послідовно в співпрограми, а базова бібліотека буде її реалізовувати асинхронно для нас [12].

Отримані після класифікації та розпізнавання данні при передачі на сервер переводяться у JSON. JSON (скорочення для JavaScript Object Notation) являє собою схематичне текстове представлення структурованих даних, яке базується на парах ключових значень та упорядкованих списках. Хоча JSON походить від JavaScript, він підтримується як локально, так і через бібліотеки більшості основних мов програмування. JSON зазвичай,

але не виключно, використовується для обміну інформацією між веб-клієнтами та веб-серверами [13].

Сховище даних – нереляційна база даних Realm. База даних Realm є альтернативою SQLite і Core Data. Realm значно швидший, ніж ORM, і зазвичай швидший, ніж SQLite. Цю базу данних активно використовують такі компанії як Netflix і Starbucks. Переваги Realm:

– простота в роботі – Realm працює з живими об’єктами і розробнику не потрібно хвилюватись, за структуру БД чи назви таблиць. Real автоматично створює документ для кожного класу, і поміщає у нього серілізовані об’єкти даного класу. Усі CRUD операції проводяться не написанням запитів, а викликом автозгенерованих методів;
– з позиції продуктивності було доведено що мобільна база даних синхронізує об'єкти значно швидше, ніж Core Data, і здійснює паралельний доступ до даних без проблем. Це означає, що кілька джерел можуть отримати доступ до одного і того ж об'єкту без необхідності керувати блокуванням або будь-яких проблем з неузгодженістю даних;
– мобільна база даних Realm пропонує служби шифрування для захисту бази на диску за допомогою AES-256 + SHA2 64-розрядного шифрування;
– база даних Realm швидше альтернативних ORM на Android приблизно в 100 разів (у порівнянні з SQLite - в 10 разів). А швидкість взаємодії з БД безпосередньо впливає на час відгуку програми [14].

При створенні андроїд додатку буде використано інтегроване середовище розробки Android Studio – офіційний засіб розробки під ОС Android. Який є зручним та інтуітивно зрозумілим інструментом для розробки під операційну систему андроїд.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Даний програмний продукт являє собою додаток для класифікації документів та переведення необхідних в залежності від типу документу даних в готову для обробки та зберігання структуру даних.

Для правильної роботи застосування до складу технічних засобів повинні входити певні компоненти.

Смартфон, що має конфігурацію наведену нижче:

- процесор з тактовою частотою не нижче 2 ГГц;
- об’єм оперативної пам’яті не менше 2048 Мб;
- наявність швидкісного інтернет-з’єднання;
- камера з розширенням матриці не менше 6 Мпікс;
- версія операційної системи вище версії Android 5.1.

4.3 Архітектура програмного забезпечення

Для даного застосування була обрано архітектурний шаблон MVVM(Model View Viewmodel). Це шаблон архітектури клієнтських додатків, який був запропонований Джоном Госсманом (John Gossman) як альтернатива шаблонами MVC і MVP при використанні технології зв'язування даних (Data Binding). Його концепція полягає в відділенні логіки представлення даних від бізнес-логіки шляхом винесення її в окремий клас для більш чіткого розмежування [15].

Вона складається з трьох компонентів:

- view – власне, це і є макет екрану, в якому розташовуються всі необхідні віджети для відображення інформації;
- model – логіка роботи з даними, що використовує додаток;
- viewModel – об'єкт, в якому описується логіка поведінки View в залежності від результату роботи Model. Можна назвати його

моделлю поведінки View. Це може бути як форматування тексту, так і логіка управління видимістю компонентів або відображення станів, таких як завантаження, помилка, порожні екрани. Також в ній описується поведінка, яке було ініційовано користувачем (введення тексту, натискання на кнопку і т.п.).

MVVM архітектура має ряд переваг на основі, яких і був зроблений вибір у її користь:

- тестування. Така структура спрощує написання тестів і процес створення mock-об'єктів. Mock-об'єкт - це інсталяції специфічної для тесту версії програмного компонента (як правило, класу), яка забезпечує звичайні результати поведінки, а також часто перевіряє, чи викликається об'єкт, що тестується. Також, в більшості випадків відпадає потреба в автоматизованому UI-тестуванні, тому що можна обернути unit-тестами сам ViewModel;
- розмежування логіки. За рахунок більшого розмежування код стає більш гнучким і простим у підтримці, не кажучи про його читабельності. Кожен модуль відповідає за свою конкретну функцію і тільки.

На рисунку 4.1 зображена модель MVVM для андроїд додатку.

Рисунок 4.1 - Модель MVVM архітектури

CRUD operations – (акронім з англійської, create read update delete) визначає чотири базові функції управління даними «створення, зчитування, редагування і видалення». Використовується для опису конфігурування користувальницького інтерфейсу, що полегшує перегляд, пошук та редагування інформації [16].

4.3.1 Діаграма класів

Діаграма класів зазвичай при моделюванні об’єктно-орієнтованих систем для проектування словника системи чи кооперацій та систем. Діаграми класів мовою UML використовують для того, щоб показати параметри блоків цих діаграм та їх зв’язків. На діаграмах класів зазвичай представлені класи, інтерфейси, залежності, узагальнення та асоціації. Також діаграми можуть включати в себе пакети або підсистеми; ті та інші групують елементи моделі у більш великі утворення.

Було створено діаграму для архітектурного шаблону проектування MVVM, який використовується у застосуванні.

Схема структурна класів програмного забезпечення наведена в частині графічного матеріалу.

4.3.2 Діаграма послідовності

Діаграма послідовності призначена для моделювання та наочного відображення сценарію роботи застосування.

Робота додатку відбувається наступним чином. Користувач заходить у додаток, обирає уже існуюче зображення або робить фото документу. Після цього користувач натискає на кнопку «Обробити». Додаток визначає клас документу, розпізнає текст та витягує з тексту необхідну для даного типу інформацію. Якщо розпізнавання пройшло успішно, додаток формує структуру даних та записує її в базу даних і після запису відображає результат користувачу. В іншому випадку додаток виводить користувачу повідомлення з причиною помилки.

Структурна схема послідовності наведена в частині графічного матеріалу.

4.3.3 Діаграма компонентів

Діаграма компонентів наочно показує розбиття програмної частини застосування на структурні компоненти та їх залежність один з одним. Компоненти поєднані за допомогою зв’язків, які показують, що один із компонентів надає сервіс, необхідний іншому компоненту. Дана діаграма забезпечує узгоджений перехід від логічного до фізичного представлення застосування у вигляді програмних компонентів.

Як можна побачити на діаграмі, застосування має п’ять інтерфейсів:

– bindData – інтерфейс відображення.
– observe – інтерфейс, що слідкує за змінами viewmodel та відображає їх відповідно.

– CRUD operations – (акронім з англійської, create read update delete) визначає чотири базові функції управління даними «створення, зчитування, редагування і видалення». Використовується для збереження та відображення.
– process data – інтерфейс, що здійснює обробку наданих користувачем зображень.

Схема структурна компонентів програмного забезпечення наведена в частині графічного матеріалу.

4.3.4 Специфікація функцій

Для предметної області автоматизації формування асортименту торговельної організації було розроблено методи для кожного класу.

В таблицях 4.1-4.4 наведено опис методів деяких класів. Опис складається із:

- назви методу;
- змістовного опису призначення методу;
- параметрів, які приймає метод, якщо вони наявні;
- змістовного опису параметрів, якщо вони наявні.

Таблиця 4.1 – Методи класу FeatureExtraction

Метод	Опис методу	Параметр	Опис параметру
getInstance()	Генерує екземпляр класу, або повертає посилання на існуючий екземпляр класу	-	-
extractFeatureStats()	Створює об'єкт FeatureStats з показниками появи ключових слів у категоріях, категорій і загальне число спостережень. Ці статистичні дані використовуються алгоритмом вибору ознак	document	Розпізнаний текст документу

Таблиця 4.2 – Методи класу NaiveBayes

Метод	Опис методу	Параметр	Опис параметру
preprocessDataset()	Попередньо обробляє оригінальний набір даних і перетворює його в список документів	map	Мапа назв документів та масиву зчитаних рядків кожного документу
predict()	Передбачає категорію тексту за допомогою вже підготовленого класифікатора і повертає його категорію	document	Об’єкт, що містить у собі розпізнаний текст
loadKnolageBase()	Завантажує базу знань	knolagebase	Файл, що містить базу знань

Таблиця 4.3 – Методи класу ProcessingViewModel

Метод	Опис методу	Параметр	Опис параметру
process()	Передає на текст на обробку об’єкту Processor та повертає результат після обробки	text	Розпізнаний текст у вигляді масиву рядків
setDataState()	Змінює значення Livedata на потрібний стан	state	Стан відображення данних
getHistoryOfSucces()	Повертає історію успішно розпізнаних документів	fromStart	Чи потрібно завантажувати заново чи наступну сторінку

Продовження таблиці 4.3

getErrorsHistory()	Повертає історію не розпізнаних	fromStart	Чи потрібно завантажувати

	документів		заново чи наступну сторінку
storeInRepository()	Додає оброблений результат в базу даних	result	Результат обробки
extractText()	Розпізнає текст та повертає його у вигляді масиву рядків	image	Зображення документу

Таблиця 4.4 – Методи класу HomeActivity

Метод	Опис методу	Параметр	Опис параметру
observe()	Підписується на Livedata свого ViewModel	-	-
handleState()	Обробляє отриманий за підпискою DataState	dataState	Поточний стан що містить в собі данні чи помилку
showData()	Відображає дані	Name, surname, email, password, shop_type	Ім’я, прізвище, електронна пошта, пароль та напрямленість магазину користувача
shoeError()	Відображає помилку	error	Помилка отримана при обробці
showLoading()	Відображає лоадер	-	-

Продовження таблиці 4.4

setupUI()	Ініціалізація компонентів	-	-

	інтерфейсу користувача		
dispatchPhotoIntent()	Відкриття камери	-	-
loadFromDevice()	Завантаження зображення з девайсу	-	-
process()	Обробка зображення	image	Зображення документу

Висновок до розділу

У цьому розділі описані засоби розробки, які були обрані для застосування інформаційної підтримки розпізнавання документів для мобільної платформи.

Наведені вимоги до технічного забезпечення, необхідного для коректного функціонування застосування.

Представлена модель архітектури програмного забезпечення, у даному випадку Model-View-ViewModel.

Наведено детальний опис діаграми класів та діаграми послідовності, що демонструють наявні в застосуванні класи та порядок їх взаємодії із базою даних та користувачем.

За допомогою діаграми компонентів показане розбиття програмної частини застосування на структурні компоненти та їх залежність один з одним.

Наведена специфікація методів, які були використані у застосуванні.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для того, щоб відкрити додаток його спочатку потрібно встановити. Завантажте .apk файли за посиланням (посилання на драйв) та відкрийте його. Відкриється системний екран інсталяції.

На даному екрані натисніть кнопку «Встановити» та дочекайтесь кінця інсталяції.

Після того, як додаток буде встановлений знайдіть іконку додатку в системному меню та натисніть на неї – відкритий головний екран додатку.

На цьому екрані відображений список розпізнаних документів описаних наступним чином зліва елемент списку розпізнаних документів містить назву файлу з зображенням, який був розпізнаний, дату, коли був розпізнаний документ, а зправа класифікований тип розпізнаного документу. Щоб переглянути результат розпізнавання будь якого елементу списку потрібно просто натиснути на нього.

Якщо система ще не розпізнала жодного документу список буде порожній, а на його місці буде відображено відповідне повідомлення.

У нижньому правому кутку є плаваюча кнопка дій «Додати», натиск на яку відкриває меню завантаження зображення

Рисунок 5.1 – Головна сторінка сайту

Натисніть на кнопку «Додати». Відкриється меню завантаження зображення (рисунок 5.2).

Рисунок 5.2 – Меню завантаження

Натисніть на кнопку «Завантажити з пристрою», щоб перейти до обрання зображення з галереї (рисунок 5.3) або «Зробити фото», щоб перейти до камери рисунок (5.4).

Рисунок 5.3 – Екран обрання зображення з пристрою

На екрані обрання фото з галереї оберіть фото документу, як показано на рисунку 5.3 та підтвердіть обрання». Відкриється екран попереднього перегляду та обробки зображення (рисунок 5.5), у якому потрібно продовжити обробку або відмінити (рисунок 5.6).

Рисунок 5.4 – Екран камери

Зробіть фото документу – додаток відкриє екран попереднього перегляду та обробки зображення (рисунок 5.5).

Рисунок 5.5 – Екран попереднього перегляду зображення

Якщо натиснути на кнопку вімінити ви повернетесь на домашній екран додатку (рисунок 5.1). Для обробки зображення натисніть кнопку «Обробити». Після натиску на кнопку система почне розпізнавати зображення, доки процес не буде завершено на екрані буде відображений лоадер (рисунок 5.5)

Рисунок 5.6 – Лоадер обробки

Як тільки зображення буде оброблене ми отримаємо повідомлення про успішну обробку або помилку та відкриється домашня екран додатку. У разі успішної обробки у списку оброблених документів буде відображений новий запис (рисунок 5.7).

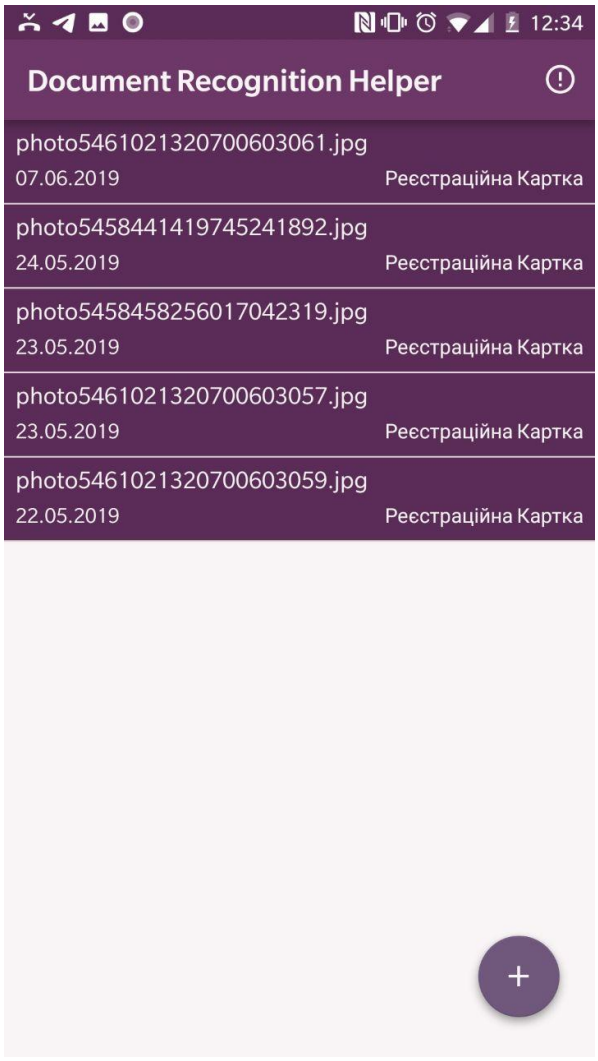


Рисунок 5.7 – Список оброблених документів

Щоб переглянути результат обробки натисніть на новий запис (рисунок 5.8).

Рисунок 5.8 – Перегляд результату обробки

Результат обробки поданий у форматі JSON. У разі отримання повідомлення про помилку після етапу обробки у історії помилок з’явиться нова помилка (рисунок 5.9). Щоб перейти в історію помилок натисніть на знак оклику на головному екрані в верхньому навігаційному меню

Рисунок 5.9 – Перегляд помилок

5.2 Випробування програмного продукту

У цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення застосування платформи функціональним вимогам, представленим у технічному завданні на створення застосування інформаційної підтримки розпізнавання документів для мобільної.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій застосування інформаційної підтримки розпізнавання документів для мобільної вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

У процесі тестування була перевірена основна функціональність застосування інформаційної підтримки розпізнавання документів для мобільної платформи. У таблицях 5.1 – 5.6 наведений перелік випробувань основних функціональних можливостей, згідно зі сценаріями діаграми використання.

Таблиця 5.1 – Тест завантаження застосування

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Завантажте застосування за посиланням	Застосування завантажене та встановлене на мобільний пристрій	пройдений
Кроки тесту		
Натисніть на іконку застосування у меню свого мобільного пристрою	Застосування відкрите і готове до роботи	пройдений
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Таблиця 5.2 – Тест обрання фото з галереї

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Кроки тесту		
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото документу, який потрібно розпізнати	Зображення готове до розпізнавання	пройдений
Післяумова		
Натисніть на кнопку «Відмінити» знизу зображення.	Відкрита головна сторінка	пройдений

Таблиця 5.3 – Тест завантаження фотографії документу

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Кроки тесту		
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Сфотографувати документ»	Відкрита камера	пройдений
Сфотографуйте документ, який потрібно розпізнати	Зображення сфотографоване	
Натисніть на галочку, щоб підтвердити фотографію	Зображення готове до розпізнавання	
Післяумова		
Натисніть на кнопку «Відмінити» знизу зображення.	Відкрита головна сторінка	пройдений

Таблиця 5.4 – Тест розпізнавання документу

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкрита головна сторінка застосування	пройдений
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото документу, який потрібно розпізнати	Зображення готове до розпізнавання	пройдений

Продовження таблиці 5.4

Кроки тесту		
Натисніть на кнопку «Обробити»	З’явився лоудер, що показує поточний процесс розпізнавання. Документ розпізнано, виведений попередній текст	пройдений
Натисніть на кнопку «Ок»	Відкритий головний екран, в історії обробки з’явився новий запис	пройдений
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Таблиця 5.5 – Тест перегляд помилок

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото, на якому немає тексту	Зображення готове до розпізнавання	пройдений
Кроки тесту		
Натисніть на кнопку «Обробити»	З’явився лоудер, що показує поточний процесс розпізнавання. Фото не було розпізнано, виведене повідомлення про помилку	пройдений
Натисніть на кнопку «Ок»	Відкритий головний екран застосування	пройдений
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Таблиця 5.6 – Тест перегляду історії помилок

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото, на якому немає тексту	Зображення готове до розпізнавання	пройдений
Кроки тесту		
Натисніть на кнопку «Обробити»	З’явився лоудер, що показує поточний процесс розпізнавання. Фото не було розпізнано, виведене повідомлення про помилку	пройдений
Натисніть на кнопку «Ок»	Відкритий головний екран застосування	пройдений
Натисніть на іконку «Помилки» у верхній навігаційній панелі	Відкритий екран з історією помилок	пройдений
Натисніть на помилку зі списку	Відкрито діалогове вікно з деталями про помилку	пройдений
Натисніть на кнопку «Ок»	Діалогове вікно зачинено	пройдений
Натиснітьв навігаційні панелі на кнопку «Назад»	Відкритий головний екран застосування	
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Висновок до розділу

У даному розділі були описані інструкції користувача по роботі із застосуванням. Детально описано, як завантажити та встановити додаток, як працювати із основною функцією застосування – розпізнаванням документів. Продемонстровано як завантажити зображення або зробити нову фотографію документу, а також як переглядати інформацію про помилки.

Неведені мета випробовувань та загальні положення для застосування інформаційної підтримки розпізнавання документів для мобільної платформи.

Розроблені та успішно пройдені тестові сценарії.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі були розглянуті основні процеси застосування інформаційної підтримки розпізнавання документів для мобільної платформи.

Розділ загальних положень надає інформацію про предметне середовище роботи. Описані діючі актори системи та їх функції. Проведено аналіз аналогів застосування.

Наведено опис інформаційного забезпечення до програмного продукту, що описує приклад сутностей нереляційної бази даних, вхідні та вихідні дані.

У розділі програмної та технічної підтримки описані технології, які використані в програмному продукті. Обґрунтовуються технології та їх особливості, які використовуються в даній розробці. В якості БД була використана Realm, а інструментом розробки вибрана мова програмування Kotlin. Наведені мінімальні вимоги до технічного забезпечення. Розглядаються описи класів, на яких базується архітектура розробки.

У технологічному розділі було наведено керівництво користувача в якому описуються детальні інструкції до розробленого застосування. Описані методи випробування на основі тестових сценаріїв.

Дане застосування автоматизує процес інформаційної підтримки розпізнавання документів для мобільної платформи.

.

ПЕРЕЛІК ПОСИЛАНЬ

1. Слободяник М. Структура сучасного документознавства // Вісн. Кн. палати. - 2003. - № 4
2. Шкилев В.Д., Желева Н.А. О некоторых особенностях восприятия электронного документа. В кн.: Перспективные методы и технологии, электронные коллекции: материалы, Дубна, 7 - 11 октября 2008 г. - 55с.
3. Пескова О.В. Автоматическое формирование рубрикатора полнотекстовых документов. В кн.: Перспективные методы и технологии, электронные коллекции: материалы, Дубна, 7 - 11 октября 2008 г. - 139с.
4. В.О. Козел. Методи та етапи автоматичного розпізнавання тексту // Вісник Черкаського університету (науковий журнал) Випуск 172. Серія прикладна математика. Інформатика С.75-86
5. Bishop C. Pattern Recognition and Machine Learning / Christopher Bishop., 2014.
6. Розробка документів із когнітивним захопленням [Електронний ресурс] // Режим доступу: https://www.abbyy.com/uk-ua/flexicapture/
7. Tesseract OCR. An optical character recognition engine [Електронний ресурс] // Режим доступу: https://opensource.google.com/projects/tesseract
8. Електронний документообіг. АЦСК «Україна» [Електронний ресурс] // Режим доступу: https://www.uakey.com.ua/index.php?num_text=7365&lang=ukr
9. Наївний баєсів класифікатор [Електронний ресурс] // Режим доступу: https://uk.unionpedia.org/i/%D0%9D%D0%B0%D1%97%D0%B2%D0%BD%D0%B8%D0%B9_%D0%B1%D0%B0%D1%94%D1%

81%D1%96%D0%B2_%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%82%D0%BE%D1%80

10. Рекомендательные системы: что это? [Електронний ресурс] // Режим доступу: https://lpgenerator.ru/blog/2015/12/25/rekomendatelnye-sistemy-chto-eto/#podrob
11. Згладжування Лапласа [Електронний ресурс] // Режим доступу: https://habr.com/ru/post/415963/
12. Kotlin vs Java: [2019] Most Important Differences That You Must Know [Електронний ресурс] // Режим доступу: https://hackr.io/blog/kotlin-vs-java
13. Introducing JSON [Електронний ресурс] // Режим доступу: https://www.json.org/
14. Realm Database [Електронний ресурс] // Режим доступу: https://realm.io/products/realm-database/
15. Model-View-ViewModel (MVVM) Explained [Електронний ресурс] // Режим доступу: https://www.wintellect.com/model-view-viewmodel-mvvm-explained/
16. What are CRUD Operations: How CRUD Operations Work, Examples, Tutorials & More [Електронний ресурс] // Режим доступу: https://stackify.com/what-are-crud-operations/

Додаток А

Тексти програмного коду

Інформаційна підтримка розпізнавання документів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк, 204 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

MainActivity.kt

```
package com.example.diploma

import android.content.Intent
import android.graphics.BitmapFactory
import android.net.Uri
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.os.Environment
import android.provider.MediaStore
import android.support.v4.content.FileProvider
import android.view.View
import kotlinx.android.synthetic.main.activity_main.*
import java.io.File
import java.io.IOException
import java.text.SimpleDateFormat
import java.util.*
import android.app.ProgressDialog
import android.support.v7.app.AlertDialog
import android.support.v7.widget.DividerItemDecoration
import android.support.v7.widget.LinearLayoutManager
import kotlin.collections.ArrayList


class MainActivity : AppCompatActivity() {
    val REQUEST_IMAGE_CAPTURE = 1
    val GALLERY_REQUEST_CODE = 2
    val REQUEST_TAKE_PHOTO = 1
    var currentPhotoPath: String = ""
    val vm: MainViewModel by lazy { MainViewModel(this) }
    lateinit var uri: Uri
    val list = ArrayList<Document>()
    lateinit var loadingDialog: ProgressDialog
    lateinit var errorDialog: AlertDialog

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setActions()
        initializeToolbar()
        initializeRecycler()
        initialState()
        observe()
        vm.loadFromRepository()
    }

    private fun setActions() {
        photo.setOnClickListener {
            dispatchTakePictureIntent()
        }
        upload.setOnClickListener {
            pickFromGallery()
        }
        cancel_action.setOnClickListener {
            initialState()
        }
        process_action.setOnClickListener {
            vm.proces(uri)
        }

    }
```

```
    private fun initializeToolbar() {
        toolbar.inflateMenu(R.menu.menu)
        toolbar.setOnMenuItemClickListener {
            startActivity(Intent(this, ErrorActivity::class.java))
            true
        }
    }

    private fun initializeRecycler() {
        rv.layoutManager = LinearLayoutManager(this)
        rv.addItemDecoration(
            DividerItemDecoration(
                rv.getContext(),
                (rv.layoutManager as LinearLayoutManager).getOrientation()
            )
        )
        rv.adapter = DocAdapter(list)
    }

    private fun pickFromGallery() {
        preprocessState()
        val intent = Intent(Intent.ACTION_PICK)
        intent.type = "image/*"
        val mimeTypes = arrayOf("image/jpeg", "image/png")
        intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes)
        startActivityForResult(intent, GALLERY_REQUEST_CODE)
    }

    @Throws(IOException::class)
    private fun createImageFile(): File {
        // Create an image file name
        val timeStamp: String =
SimpleDateFormat("yyyyMMdd_HHmmss").format(Date())
        val storageDir: File =
getExternalFilesDir(Environment.DIRECTORY_PICTURES)
        return File.createTempFile(
            "JPEG_${timeStamp}_", /* prefix */
            ".jpg", /* suffix */
            storageDir /* directory */
        ).apply {
            // Save a file: path for use with ACTION_VIEW intents
            currentPhotoPath = absolutePath
        }
    }


    private fun dispatchTakePictureIntent() {
        Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
            // Ensure that there's a camera activity to handle the intent
            takePictureIntent.resolveActivity(packageManager)?.also {
                // Create the File where the photo should go
                val photoFile: File? = try {
                    createImageFile()
                } catch (ex: IOException) {
                    // Error occurred while creating the File
                    null
                }
                // Continue only if the File was successfully created
                photoFile?.also {
                    val photoURI: Uri = FileProvider.getUriForFile(
                        this,
                        "com.example.diploma.fileprovider",
                        it
```

```
                    )
                    uri = photoURI
                    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
photoURI)
                    startActivityForResult(takePictureIntent,
REQUEST_TAKE_PHOTO)
                }
            }
        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK)
{
            preprocessState()
            setPic()
        }
        if (requestCode == GALLERY_REQUEST_CODE && resultCode == RESULT_OK) {
            preprocessState()
            val selectedImage = data?.data
            uri = selectedImage!!
            imageView.setImageURI(selectedImage)
        }
    }

    private fun setPic() {
        // Get the dimensions of the View
        val targetW: Int = imageView.width
        val targetH: Int = imageView.height

        val bmOptions = BitmapFactory.Options().apply {
            // Get the dimensions of the bitmap
            inJustDecodeBounds = true
            BitmapFactory.decodeFile(currentPhotoPath, this)
            val photoW: Int = outWidth
            val photoH: Int = outHeight
            val scaleFactor: Int = Math.min(photoW / targetW, photoH /
targetH)

            inJustDecodeBounds = false
            inSampleSize = scaleFactor
            inPurgeable = true
        }
        BitmapFactory.decodeFile(currentPhotoPath, bmOptions)?.also { bitmap
->
            imageView.setImageBitmap(bitmap)
        }
    }

    private fun observe() {
        vm.stateLiveData.observe(this, android.arch.lifecycle.Observer {
            when (it?.state) {
                State.DATA -> {
                   .document?.let { it1 -> showData(it1) }
                    errorState(false)
                    loadingState(false)
                }
                State.ERROR -> {
                    errorState(true)
                    loadingState(false)
                }
```

```
                State.LOADING -> {
                    errorState(false)
                    loadingState(true)
                }

            }
        })
    }

    fun loadingState(visible: Boolean) {
        if (visible) {
            loadingDialog = ProgressDialog(this).apply {
                setMessage("Processing ...")
            }
        } else
            loadingDialog.dismiss()
    }

    fun showData(list: List<Document>) {
        rv.adapter = DocAdapter(list as ArrayList<Document>)
    }

    fun errorState(visible: Boolean) {
        if (visible) {
            errorDialog = AlertDialog.Builder(this).apply {
                setMessage("Error occured check errors history")
            }.show()
        } else
            errorDialog.dismiss()

    }

    fun initialState() {
        rv.visibility = View.VISIBLE
        image_layout.visibility = View.INVISIBLE
        multiple_actions.visibility = View.VISIBLE
    }

    fun preprocessState() {
        rv.visibility = View.INVISIBLE
        image_layout.visibility = View.VISIBLE
        multiple_actions.visibility = View.INVISIBLE
    }
}
```

FeatureExtraction.kt

```
package com.example.diploma


import com.example.diploma.processor.Document
import com.example.diploma.processor.FeatureStats
import java.util.HashMap

class FeatureExtraction {


    fun extractFeatureStats(dataset: List<Document>): FeatureStats {
        val stats = FeatureStats()

        var categoryCount: Int?
        var category: String
        var featureCategoryCount: Int?
        var feature: String
```

```
        var featureCategoryCounts: Map<String, Int>?
        for (doc in dataset) {
            ++stats.n
            category = doc.category


            categoryCount = stats.categoryCounts[category]
            if (categoryCount == null) {
                stats.categoryCounts[category] = 1
            } else {
                stats.categoryCounts[category] = categoryCount + 1
            }

            for ((key) in doc.tokens) {
                feature = key

                featureCategoryCounts =
stats.featureCategoryJointCount[feature]
                if (featureCategoryCounts == null) {
                    stats.featureCategoryJointCount[feature] = HashMap()
                }

                featureCategoryCount =
stats.featureCategoryJointCount[feature]!![category]
                if (featureCategoryCount == null) {
                    featureCategoryCount = 0
                }

                stats.featureCategoryJointCount[feature]!![category] =
++featureCategoryCount
            }
        }

        return stats
    }


    fun chisquare(stats: FeatureStats, criticalLevel: Double): Map<String,
Double> {
        val selectedFeatures = HashMap<String, Double>()

        var feature: String
        var category: String
        var categoryList: Map<String, Int>

        var N1dot: Int
        var N0dot: Int
        var N00: Int
        var N01: Int
        var N10: Int
        var N11: Int
        var chisquareScore: Double
        var previousScore: Double?
        for ((key, value) in stats.featureCategoryJointCount) {
            feature = key
            categoryList = value

            N1dot = 0
            for (count in categoryList.values) {
                N1dot += count!!
            }

            N0dot = stats.n - N1dot
```

```
            for ((key1, value1) in categoryList) {
                category = key1
                N11 = value1
                N01 = stats.categoryCounts[category]!! - N11

                N00 = N0dot - N01
                N10 = N1dot - N11

                chisquareScore = stats.n * Math.pow(
                    (N11 * N00 - N10 * N01).toDouble(),
                    2.0
                ) / ((N11 + N01) * (N11 + N10) * (N10 + N00) * (N01 + N00))

                if (chisquareScore >= criticalLevel) {
                    previousScore = selectedFeatures[feature]
                    if (previousScore == null || chisquareScore >
previousScore) {
                        selectedFeatures[feature] = chisquareScore
                    }
                }
            }
        }

        return selectedFeatures
    }
}
```

NaïveBayes.kt

```
package com.example.diploma.processor

import java.util.ArrayList
import java.util.HashMap

class NaiveBayes @JvmOverloads constructor(knowledgeBase:
NaiveBayesKnowledgeBase? = null) {
    var chisquareCriticalValue = 10.83

    var knowledgeBase: NaiveBayesKnowledgeBase? = null
        private set

    init {
        this.knowledgeBase = knowledgeBase
    }

    private fun preprocessDataset(trainingDataset: Map<String,
Array<String>>): List<Document> {
        val dataset = ArrayList<Document>()

        var category: String
        var examples: Array<String>

        var doc: Document

        val it = trainingDataset.entries.iterator()

        while (it.hasNext()) {
            val entry = it.next()
            category = entry.key
            examples = entry.value
```

```
            for (i in examples.indices) {
                doc = TextTokenizer.tokenize(examples[i])
                doc.category = category
                dataset.add(doc)
            }

        }

        return dataset
    }

    private fun selectFeatures(dataset: List<Document>): FeatureStats {
        val featureExtractor = FeatureExtraction()

        val stats = featureExtractor.extractFeatureStats(dataset) //extract
the stats of the dataset

        val selectedFeatures = featureExtractor.chisquare(stats,
chisquareCriticalValue)

        val it = stats.featureCategoryJointCount.entries.iterator()
        while (it.hasNext()) {
            val feature = it.next().key

            if (selectedFeatures.containsKey(feature) == false) {
                it.remove()
            }
        }

        return stats
    }

    @Throws(IllegalArgumentException::class)
    fun train(trainingDataset: Map<String, Array<String>>, categoryPriors:
Map<String, Double>?) {
        val dataset = preprocessDataset(trainingDataset)

        val featureStats = selectFeatures(dataset)


        knowledgeBase = NaiveBayesKnowledgeBase()
        knowledgeBase!!.n = featureStats.n //number of observations
        knowledgeBase!!.d = featureStats.featureCategoryJointCount.size
//number of features


        if (categoryPriors == null) {
            knowledgeBase!!.c = featureStats.categoryCounts.size //number of
cateogries
            knowledgeBase!!.logPriors = HashMap()

            var category: String
            var count: Int
            for ((key, value) in featureStats.categoryCounts) {
                category = key
                count = value

                knowledgeBase!!.logPriors[category] =
Math.log(count.toDouble() / knowledgeBase!!.n)
            }
        } else {
            knowledgeBase!!.c = categoryPriors.size
```

```
            if (knowledgeBase!!.c != featureStats.categoryCounts.size) {
                throw IllegalArgumentException("Invalid priors Array: Make
sure you pass a prior probability for every supported category.")
            }

            var category: String
            var priorProbability: Double?
            for ((key, value) in categoryPriors) {
                category = key
                priorProbability = value
                if (priorProbability == null) {
                    throw IllegalArgumentException("Invalid priors Array:
Make sure you pass a prior probability for every supported category.")
                } else if (priorProbability < 0 || priorProbability > 1) {
                    throw IllegalArgumentException("Invalid priors Array:
Prior probabilities should be between 0 and 1.")
                }

                knowledgeBase!!.logPriors[category] =
Math.log(priorProbability)
            }
        }

        val featureOccurrencesInCategory = HashMap<String, Double>()

        var occurrences: Int?
        var featureOccSum: Double?
        for (category in knowledgeBase!!.logPriors.keys) {
            featureOccSum = 0.0
            for (categoryListOccurrences in
featureStats.featureCategoryJointCount.values) {
                occurrences = categoryListOccurrences[category]
                if (occurrences != null) {
                    featureOccSum += occurrences.toDouble()
                }
            }
            featureOccurrencesInCategory[category] = featureOccSum
        }

        var feature: String
        var count: Int?
        var featureCategoryCounts: Map<String, Int>
        var logLikelihood: Double
        for (category in knowledgeBase!!.logPriors.keys) {
            for ((key, value) in featureStats.featureCategoryJointCount) {
                feature = key
                featureCategoryCounts = value

                count = featureCategoryCounts[category]
                if (count == null) {
                    count = 0
                }

                logLikelihood = Math.log((count + 1.0) /
(featureOccurrencesInCategory[category]!! + knowledgeBase!!.d))
                if (knowledgeBase!!.logLikelihoods.containsKey(feature) ==
false) {
                    knowledgeBase!!.logLikelihoods[feature] = HashMap()
                }
                knowledgeBase!!.logLikelihoods[feature]!![category] =
logLikelihood
            }
        }
```

```
    }


    @Throws(IllegalArgumentException::class)
    fun predict(text: String): String? {
        if (knowledgeBase == null) {
            throw IllegalArgumentException("Naive Bayes cannot work without
knowledge base")
        }

        val doc = TextTokenizer.tokenize(text)


        var category: String
        var feature: String
        var occurrences: Int?
        var logprob: Double?

        var maxScoreCategory: String? = null
        var maxScore: Double? = java.lang.Double.NEGATIVE_INFINITY

        for ((key, value) in knowledgeBase!!.logPriors) {
            category = key
            logprob = value

            for ((key1, value1) in doc.tokens) {
                feature = key1

                if (!knowledgeBase!!.logLikelihoods.containsKey(feature)) {
                    continue
                }

                occurrences = value1

                logprob += occurrences!! *
knowledgeBase!!.logLikelihoods[feature]!![category]!!
            }

            if (logprob > maxScore) {
                maxScore = logprob
                maxScoreCategory = category
            }
        }

        return maxScoreCategory
    }
}

ActivityError.kt
package com.example.diploma

import android.app.ProgressDialog
import android.os.Bundle
import android.support.v7.app.AlertDialog
import android.support.v7.app.AppCompatActivity
import android.support.v7.widget.DividerItemDecoration
import android.support.v7.widget.LinearLayoutManager
import kotlinx.android.synthetic.main.activity_error.*
import java.util.ArrayList

class ErrorActivity : AppCompatActivity() {

    lateinit var loadingDialog: ProgressDialog
```

```
    lateinit var errorDialog: AlertDialog
    val vm: ErrorViewModel by lazy { ErrorViewModel(this) }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_error)
        rv.layoutManager = LinearLayoutManager(this)
        rv.addItemDecoration(
            DividerItemDecoration(
                rv.getContext(),
                (rv.layoutManager as LinearLayoutManager).getOrientation()
            )
        )
        toolbar.setTitle("Errors")
        observe()
    }

    private fun observe() {
        vm.stateLiveData.observe(this, android.arch.lifecycle.Observer {
            when (it?.state) {
                State.DATA -> {
                    it.document?.let { it1 -> showData(it1) }
                    errorState(false)
                    loadingState(false)
                }
                State.ERROR -> {
                    errorState(true)
                    loadingState(false)
                }
                State.LOADING -> {
                    errorState(false)
                    loadingState(true)
                }

            }
        })
    }

    fun loadingState(visible: Boolean) {
        if (visible) {
            loadingDialog = ProgressDialog(this).apply {
                setMessage("Processing ...")
            }
        } else
            loadingDialog.dismiss()
    }

    fun showData(list: List<Document>) {
        rv.adapter = DocAdapter(list as ArrayList<Document>)
    }

    fun errorState(visible: Boolean) {
        if (visible) {
            errorDialog = AlertDialog.Builder(this).apply {
                setMessage("Error occured check errors history")
            }.show()
        } else
            errorDialog.dismiss()

    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проекту
_____ *Ю.О. Олійник*
(підпис) (ініціали, прізвище)
“15” квітня 2019 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“16” квітня 2019 р.

Інформаційна підтримка розпізнавання документів для мобільної платформи

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5212.1181-с.ТЗ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Інформаційна підтримка розпізнавання документів для мобільної платформи.

1.2 Найменування організації-замовника та організацій-учасників робіт

Генеральним замовником проекту являється кафедра Автоматизованих систем обробки інформації та управління КПІ ім. Сікорського. Представниками замовника є Олійник Юрій Олександрович.

Розробником системи є студент групи ІС-52 факультету інформатики та обчислювальної техніки КПІ ім. Ігоря Сікорського Камінський Андрій Володимирович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над застосуванням інформаційної підтримки розпізнавання документів для мобільної платформи – 5 лютого 2019 рік.

Плановий термін по закінченню роботи над застосуванням інформаційної підтримки розпізнавання документів для мобільної платформи – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ КОМПЛЕКСУ ЗАДАЧ

2.1 Призначення комплексу задач

Призначенням розробки є інформаційна підтримка розпізнавання документів для мобільної платформи, для спрощення ведення докуметнообігу.

2.2 Цілі створення комплексу задач

Головною метою розробки є спрощення процесу розпізнавання документів для переведення інформації у електронний формат.

Цілями розробки є:

- полегшення процесу ведення документообігу;
- спрощення розпізнавання однотипних документів.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- розробити функцію для розпізнавання тексту;
- розробити функцію класифікації документів;
- розробити функцію переведення тексту у структуру;
- розробити базу даних для збереження розпізнаної інформації;
- розробити функцію трекінгу помилок при обробці.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для користування застосуванням обов'язковою умовою для користувача є наявність камери.

Працювати з застосуванням можна користувачам без попередньої авторизації, користувачі мають доступ до всіх функцій розробки.

Об'єктом автоматизації є процес інформаційної підтримки розпізнавання документів для мобільної платформи.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Застосування має створювати умови для підтримки процесу розпізнавання документів. Дане застосування повинно задовольняти потреби користувачів, яким потрібна допомога у перенесенні інформації з паперових документів у електронний формат.

Застосування має виконувати наступні функції:

1. Застосування надає можливість користувачу завантаження зображення.

1.1. Застосування надає користувачу доступ до завантаження зображення з галереї.

1.2. Застосування надає можливість користувачу сфотографувати документ для обробки.

2. Застосування надає можливість користувачу переглянути попередній результат обробки.

3. Застосування надає можливість користувачу переглянути історію оброблених документів

4. Застосування надає можливість користувачу переглянути історію помилок необроблених документів

4.2 Вимоги до надійності

Вимоги до навколишнього середовища для використовування програмного забезпечення не висуваються. Помилка програмного забезпечення визначається як несподіване припинення виконання поставленого завдання без повернення результатів роботи.

Система повинна відповідати наступним вимогам:

- система та документація представляється українською мовою. Доступ до системи надається за допомогою всесвітньої мережі Інтернет або за наявності копії програмного продукту в локальному середовищі;
- усі поля вводу інформації перевіряються на відповідність необхідним даним. У разі виявлення помилки користувач буде повідомлений про помилку;
- усі дані, які згенеровані та отримані від користувача зберігаються в локальній системі для подальшого використання. Користувач має можливість видалити їх за власним бажанням.

4.3 Вимоги до складу і параметрів технічних засобів

Характеристики системи, яким повинен відповідати пристрій для коректної роботи системи для клієнта:

Мінімальні системі вимоги

Смартфон, що має конфігурацію наведену нижче:

- процесор з тактовою частотою не нижче 1 ГГц;
- об’єм оперативної пам’яті не менше 2048 Мб;
- камера з розширенням матриці не менше 6 Мпікс;
- версія операційної системи вище версії Android 5.1.

Рекомендовані системі вимоги

Смартфон, що має конфігурацію наведену нижче:

- процесор з тактовою частотою не нижче 2 ГГц;
- об’єм оперативної пам’яті не менше 2048 Мб;
- наявність швидкісного інтернет-з’єднання;
- камера з розширенням матриці не менше 12 Мпікс;
- версія операційної системи вище версії Android 6.0.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки автоматизації формування асортименту торговельної організації.

№	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	07.02.2019	
2.	Розробка сценарію роботи	12.02.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	20.02.2019	
4.	Узгодження з керівником інтерфейсу користувача	02.03.2019	
5.	Розробка інформаційного забезпечення	17.03.2019	
6.	Розробка програмного забезпечення	29.03.2019	
7.	Налагодження програми	13.04.2019	
8.	Тестування програми	27.04.2019	
9.	Здача готового програмного продукту замовнику	12.05.2019	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Для контролю коректної роботи програмного забезпечення буде проведено функціональне тестування. В ході тестування буде проведено окреме випробування основних модулів системи.

Тестування модулю пошуку полягає в написанні тестів, які перевіряють відповідність знайдених даних з вихідними даними пошуку.

Тестування інтерфейсу програми полягає в тому, що користувачу надається сторінка з відповідною формою, в яку він вводить тестові дані

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО

Керівник проекту

_____ *Ю.О. Олійник*
(підпис) (ініціали, прізвище)

“16” травня 2019 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)

“17” травня 2019 р.

Інформаційна підтримка розпізнавання документів для мобільної платформи

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5212.1181-с.ПМВ*

на 12 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Найменування програми: «Інформаційна підтримка розпізнавання документів для мобільної платформи».

1.2 Область застосування

Програма використовується для полегшення ведення документообігу, а саме для класифікації паперових документів.

1.3 Умовне позначення програми

Умовне позначення «DocumentRecognitionHelper»

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань являється перевірка відповідності функцій прикладного програмного забезпечення застосування «Інформаційної підтримки розпізнавання документів для мобільної платформи».

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Застосування має створювати умови для підтримки процесу розпізнавання документів та ведення електронного документообігу. Дане застосування повинно задовольняти потреби користувачів, яким потрібна допомога у перенесенні даних з паперових документів.

Застосування має виконувати наступні функції:

- Застосування надає можливість користувачу розпізнавати документи.
- Застосування надає користувачу можливість завантажити з галереї зображення документу для подальшого розпізнавання.
- Застосування надає можливість користувачу сфотографувати документ для подальшого розпізнавання.
- Застосування надає можливість користувачу переглянути попередній результат розпізнавання.
- Застосування надає можливість користувачу переглянути звіт про помилки в розпізнаванні документу.
- Застосування надає можливість користувачу переглянути історію розпізнаних документів.

Інтерфейс повинен бути інтуїтивно зрозумілим та зручним. Усі елементи для навігації, як кнопки та посилання повинні бути оформленні без надлишкової кількості графічних елементів та завантажуватися вчасно.

3.1.1 Вимоги до складу виконуваних функцій

Тестування виконується на підставі вимог та складу виконуваних функцій, що наведені у технічному завданні.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Керівництво користувача та код програмного продукту складають програмну документацію.

Основою для випробувань є наступні документи:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Етапи випробувань:

- ознайомчий;
- виконавчий.

На ознайомчому етапі проводиться:

- перевірка комплектності програмної документації;
- перевірка комплектності складу технічних і програмних засобів.

Під час виконавчого етапу проводиться:

- перевірка відповідності технічних характеристик системи;
- перевірка ступеню виконання вимог функціонального призначення системи.

Варіанти використання стали основою для тестування. Наведемо обрані сценарії:

- сценарій «Завантаження застосування»;
- сценарій «Обрання фото з галереї»;
- сценарій «Завантаження фотографії документу»;
- сценарій «Розпізнавання документу»;
- сценарій «Перегляд помилок»;
- сценарій «Перегляд історії помилок».

6 МЕТОДИ ВИПРОБУВАНЬ

6.1 Функціональне тестування

У процесі тестування була перевірена основна функціональність застосування інформаційної підтримки розпізнавання документів для мобільної платформи. Функціональне тестування відбувається згідно сценаріїв, що наведені у таблицях 6.1 – 6.6.

Таблиця 6.1 – Тест завантаження застосування

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Завантажте застосування за посиланням	Застосування завантажене та встановлене на мобільний пристрій	пройдений
Кроки тесту		
Натисніть на іконку застосування у меню свого мобільного пристрою	Застосування відкрите і готове до роботи	пройдений
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Таблиця 6.2 – Тест обрання фото з галереї

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Кроки тесту		
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений

Продовження таблиці 6.2

Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото документу, який потрібно розпізнати	Зображення готове до розпізнавання	пройдений
Післяумова		
Натисніть на кнопку «Відмінити» знизу зображення.	Відкрита головна сторінка	пройдений

Таблиця 6.3 – Тест завантаження фотографії документу

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Кроки тесту		
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Сфотографувати документ»	Відкрита камера	пройдений
Сфотографуйте документ, який потрібно розпізнати	Зображення сфотографоване	
Натисніть на галочку, щоб підтвердити фотографію	Зображення готове до розпізнавання	
Післяумова		
Натисніть на кнопку «Відмінити» знизу зображення.	Відкрита головна сторінка	пройдений

Таблиця 6.4 – Тест розпізнавання документу

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкрита головна сторінка застосування	пройдений
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото документу, який потрібно розпізнати	Зображення готове до розпізнавання	пройдений
Кроки тесту		
Натисніть на кнопку «Обробити»	З’явився лоудер, що показує поточний процесс розпізнавання. Документ розпізнано, виведений попередній текст	пройдений
Натисніть на кнопку «Ок»	Відкритий головний екран, в історії обробки з’явився новий запис	пройдений
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Таблиця 6.5 – Тест перегляд помилок

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений

Продовження таблиці 6.5

Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото, на якому немає тексту	Зображення готове до розпізнавання	пройдений
Кроки тесту		
Натисніть на кнопку «Обробити»	З’явився лоудер, що показує поточний процесс розпізнавання. Фото не було розпізнано, виведене повідомлення про помилку	пройдений
Натисніть на кнопку «Ок»	Відкритий головний екран застосування	пройдений
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Таблиця 6.6 – Тест перегляду історії помилок

Функція/ Use Case:		
Дія:	Очікуваний результат:	Результат тесту:
Передумова		
Відкрийте застосування	Відкритий головний екран застосування	пройдений
Натисніть на кнопку «Додати», щоб обрати спосіб подання зображення	Розгорнуті іконки «Сфотографувати документ» та «Обрати з галереї»	пройдений
Натисніть кнопку «Обрати з галереї»	Відкрита галерея	пройдений
Натисніть на фото, на якому немає тексту	Зображення готове до розпізнавання	пройдений
Кроки тесту		
Натисніть на кнопку «Обробити»	З’явився лоудер, що показує поточний процесс розпізнавання. Фото не було розпізнано, виведене повідомлення про помилку	пройдений

Продовження таблиці 6.6

Натисніть на кнопку «Ок»	Відкритий головний екран застосування	пройдений
Натисніть на іконку «Помилки» у верхній навігаційній панелі	Відкритий екран з історією помилок	пройдений
Натисніть на помилку зі списку	Відкрито діалогове вікно з деталями про помилку	пройдений
Натисніть на кнопку «Ок»	Діалогове вікно зачинено	пройдений
Натиснітьв навігаційні панелі на кнопку «Назад»	Відкритий головний екран застосування	
Післяумова		
Натисніть на кнопку «Вийти»	Застосування закрите, відкритий головний екран	пройдений

Графічний матеріал
до дипломного проекту

на тему: «Інформаційна підтримка розпізнавання документів для мобільної платформи»

Київ – 2019 року

					ДП ІС-5212.1181-с.ССД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності	Літера	Маса / Масштаб
Розробив		Камінський А.В.					
Перевірив		Олійник Ю.О.				Аркуш 1	Аркушів 1
Т. кон.					Інформаційна підтримка розпізнавання документів для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Олійник Ю.О.					

					ДП ІС-5212.1181-с.ССВ			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання			
Розробив		Камінський А.В.						
						Аркуш 1		Аркушів 1
Перевірив		Олійник Ю.О.						
Т. кон.					Інформаційна підтримка розпізнавання документів для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Н. кон.		Халус О.А.						
Затвердив		Олійник Ю.О.						

Рішення з математичного забезпечення

Схема наївного баєсовського алгоритму

КРОК 1. Визначити передумови для навчання класифікатора

КРОК 2. Обчислити розріджену матрицю термінів документів для кожного класу.

КРОК 3. Обчислити частоти, з якими зустрічається кожне слово в тексті.

КРОК 4. Обчислити $P(c|d) = \frac{P(d|c)P(c)}{P(d)}$ – ймовірність що документ d належить класу c

КРОК 5. Розрахувати ймовірність c_{map} для всіх класів і вибрати той клас, який має максимальну вірогідність за формулою з урахуванням згладжування Лапласа

$$c_{map} = \arg\max_{c \in C} \left[\log \frac{D_c}{D} + \sum_{i=1}^{n} \log \frac{W_{ic} + 1}{|V| + \sum_{i' \in V} W_{i'c}} \right]$$

Демонстраційний плакат до дипломного проекту

„Інформаційна підтримка розпізнавання документів для мобільної платформи ”

Виконав студент гр. ІС-52 Камінський А.В.

Керівник ДП Олійник Ю.О.

ДП ІС-5212.1181-с.ССК

class UML

ErrorActivity
- vm: ErrorsViewModel
- observe()
- setupUi()
- showData()
- showError()
- showLoading()

ErrorsViewModel
- repo: DocumentsRepository
- state: MutableLiveData<State>
+ getErrors()

DocumentsRepository
- realmInstance: Realm
+ addDocument(): Single<Document>
+ addError(): SingleError
+ getDocuments(): Single<List<Documents>
+ getErrors(): Single<List<Error>

TextRecognizer
+ recognize(): string

MainActivity
- vm: MainActivityViewModel
- dispachCameraIntent()
- intializeActions()
- intializeRecycler()
- intializeToolbar()
- loadFromGalery()
- observe()
- showData(List<Documents>)
- showError(Boolean)
- showLoading()

MainActivityViewModel
- processor: Processor
- repo: DocumentsRepository
+ state: MutableLiveData<State>
- textRecognizer: TextRecognizer
- addDocument()
- addError()
- classifyAndCreate(): Document
+ loadData()
- recognizeText(): string

Processor
- naiveAlgo: NaiveBayse
+ recognize(): Document

NaiveBayes
- knowledgeBase: NaiveBayesKnowledgeBase
+ getChisquareCriticalValue(): double
+ getKnowledgeBase(): NaiveBayesKnowledgeBase
+ NaiveBayes(NaiveBayesKnowledgeBase)
+ NaiveBayes()
+ predict(String): String
- preprocessDataset(Map<String, String[]>): List<Document>
- selectFeatures(List<Document>): FeatureStats
+ setChisquareCriticalValue(double): void
+ train(Map<String, String[]>, Map<String, Double>): void
+ train(Map<String, String[]>): void

-knowledgeBase

NaiveBayesKnowledgeBase
+ c: int = 0
+ d: int = 0
+ logLikelihoods: Map<String, Map<String, Double>> = new HashMap<>()
+ logPriors: Map<String, Double> = new HashMap<>()
+ n: int = 0

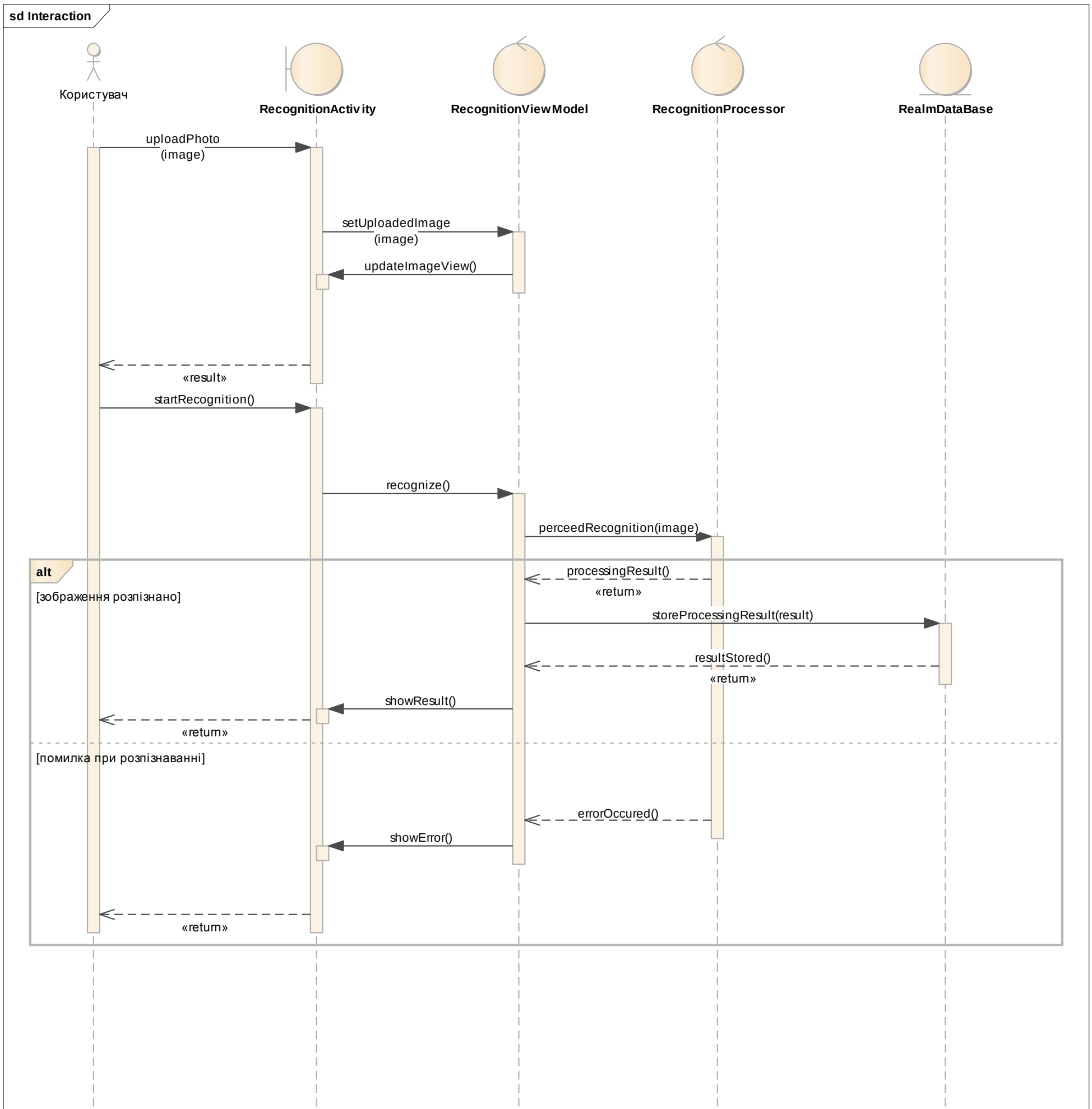
FeatureStats
+ categoryCounts: Map<String, Integer>
+ featureCategoryJointCount: Map<String, Map<String, Integer>>
+ n: int
+ FeatureStats()

processor.Document
+ category: String
+ tokens: Map<String, Integer>
+ processor.Document()

TextTokenizer
+ extractKeywords(String): String[]
+ getKeywordCounts(String[]): Map<String, Integer>
+ preprocess(String): String
+ tokenize(String): processor.Document

FeatureExtraction
+ chisquare(FeatureStats, double): Map<String, Double>
+ extractFeatureStats(List<Document>): FeatureStats

					ДП ІС-5212.1181-с.ССК	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Камінський А.В.			Схема структурна класів	Літера: — Маса: — Масштаб: —
Перевірив		Олійник Ю.О.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Халус О.А.			Інформаційна підтримка розпізнавання документів для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Затвердив		Олійник Ю.О.				



					ДП ІС-5212.1181-с.ССП	
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна послідовності	Лит. / Маса / Масштаб
Розроб.		Камінський А.В.				
Перев.		Олійник Ю.О.				Аркуш 1 / Аркушів 1
Т. Кон.					Інформаційна підтримка розпізнавання документів для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. Кон.		Халус О.А.				
Затв.		Олійник Ю.О.				

					ДП ІС-5212.1181-с.ССК		
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Камінський А.В.			Схема структурна компонентів	Літера	Маса / Масштаб
Перевірив		Олійник Ю.О.				Аркуш 1	Аркушів 1
Т. кон.					Інформаційна підтримка розпізнавання документів для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Олійник Ю.О.					